Business Process Coordination: State of the Art, Trends, and Open Issues

Umeshwar Dayal
Hewlett-Packard Labs.
Palo Alto, California
USA
dayal@hpl.hp.com

Meichun Hsu
CommerceOne Labs.
Cupertino, California
USA
meichun.hsu@commerceone.com

Rivka Ladin
Compaq Tandem Labs.
Haifa
Israel
rivka.ladin@compaq.com

Abstract

Over the past decade, there has been a lot of work in developing middleware for integrating and automating enterprise business processes. Today, with the growth in e-commerce and the blurring of enterprise boundaries, there is renewed interest in business process coordination, especially for inter-organizational processes. This paper provides a historical perspective on technologies for intra- and inter-enterprise business processes, reviews the state of the art, and exposes some open research issues. We include a discussion of process-based coordination and event/rule-based coordination, and corresponding products and standards activities. We provide an overview of the rather extensive work that has been done on advanced transaction models for business processes, and of the fledgling area of business process intelligence.

1. Introduction

Competitive pressures are forcing organizations to increasingly integrate and automate their business operations such as order processing, procurement, claims processing, administrative procedures and the like. Such *business processes* are typically of long duration, involve coordination across many manual and automated tasks, require access to several different databases and the invocation of several application systems (e.g., ERP systems), and are governed by complex business rules. A typical business process may consist of up to 100 IT transactions. Coordinating the entire process correctly and efficiently places severe demands on the organization's IT infrastructure.

In addition, with the growth of e-commerce and the trend toward increasing globalization of operations and outsourcing of functions to external service providers, there is an emerging need to integrate and automate processes that span organizational boundaries (the so-called B2C and B2B processes). One industry analyst predicts: "By 2003, more than 90 percent of e-businesses will be exploiting process automation technology" [Gartner2000]. These place additional demands on the IT infrastructure.

Different forms of middleware have been introduced to enable integration and automation of business processes, both within and across organizations. *Message brokers*, *transactional queue managers*, and *publish/subscribe mechanisms* provide means for automating processes by allowing the component applications to post events and to react to events posted by other component applications. The logic for how to react to events and chain the steps of the process typically resides in the applications themselves. This provides autonomy for the applications and flexibility in that the logic for reacting to events can be changed as business needs or policies change.

In contrast, a *business process management system* (or *business process manager*) is a middleware system that provides a central point of control for defining business processes and orchestrating their execution [WfMC, JB96, SGJR97]. The process manager records the execution state of the process and routes requests to component applications or human agents to execute tasks. Business process management systems have their roots in departmental workflow (or office automation) systems where the goal was to route work items or documents among human workers. Over the years, research prototypes and commercial systems have been developed

Proceedings of the 27th VLDB Conference,
Roma, Italy, 2001

to deal with enterprise-scale processes that include both human and automated tasks. Such systems typically provide transactional semantics and support for backward and forward recovery of business processes. Business process managers may (and, in fact, often do) rely on an underlying message broker, transactional queue manager, or publish/subscribe middleware layer to wrap participating applications, to detect business process-related events, and to guarantee reliable delivery of events and messages to applications.

In [DHL90, DHL91], we introduced two approaches to modelling and managing business processes (which we called *long-running activities*). The first approach used database rules (triggers) and transactions to model business processes [DHL90]. Subsequently, we realized that relying on triggers alone to chain the steps of a business process led to executions that might be difficult to understand or explain, because the logic was scattered over many rules. We, therefore, introduced a new model in which the "normal" logic of a business process was explicitly defined (scripted), and rules were used primarily for checking integrity constraints, handling exceptions, and so on [DHL91]. In practice, today, business processes are often modelled through a combination of scripts and rules. The script defines the process flow at a high level, and the rules are used to dynamically change the flow at execution time, determine the next step to be executed, or bind the next step to be executed to a specific resource (human or software); the decision may be based on the process execution state, other relevant data that might be retrieved from databases or passed in messages, the availability of resources, and business policies.

Over the last decade, there has been a lot more research in business process modelling, advanced transaction models for business processes, and architectures and implementation techniques for business process management systems, and numerous commercial products have appeared. There is also recent work in inter-enterprise collaborative business process management, and standards are being introduced by various consortia such as RosettaNet and ebXML.

Another recent area of research is that of *business process intelligence*. The motivation for business process automation is to improve operational efficiencies and reduce error, but commercial business process management middleware lacks tools for quantitatively tracking these business metrics. Business process intelligence aims to apply data warehousing, data analysis, and data mining techniques to process execution data, thus enabling the analysis, interpretation, and optimization of business processes.

In this paper, we provide a historical perspective and an overview of the state of the art in business process coordination, and we identify open research issues. In Section 2, we describe a framework that allows us to describe the space of models and approaches to business process management. We discuss both intra- and inter-enterprise business process management. In Section 3, we focus on publish/subscribe, messaging, and queuing services for implementing business processes. In Section 4, we discuss advanced transaction models for business processes, touching on both research and commercial practice. In Section 5, we give a brief introduction to the emerging area of business process intelligence. We conclude in Section 6 with the disclaimer that we could not possibly be comprehensive in our coverage of this vast area, so we have chosen to highlight the topics that reflect our own interests.

2. A Framework for Business Process Management

2.1 Approaches to Business Process Modelling and Coordination

A business process is a persistent unit of work started by a business event such as an invoice, request for proposal or a request for funds transfer. The process is driven by business rules that trigger tasks and sub-processes, with each state transition being executed within a transaction and audited for business reasons when required. Tasks and sub-processes are assigned to *resources*, which are organizational units that are capable and authorized to play specific *roles* in the processes.

The scripting of the rules, tasks, sub-processe, and resource policies, constitute a *process description*. An execution of a business process consists of invoking existing business services, which can reside anywhere. The context associated with the process is usually stored in a database and archived on completion. The process manager manages the state of a business process, and routes requests among participating applications.

Most commonly, the sequence of steps to be traversed in executing a business process is defined before the process instance is initiated. Most business process management systems allow branch logic to determine the next step, after a step completes, so that different sequences of steps can be followed according to the outcome of the branch logic. Often a graphical tool is used to construct the path, the decision points and the branch logic. The branch logic is usually called 'rules'. These rules are evaluated based on workflow context, workflow history, current resource availability database context, and business policies. Figure 1 illustrates a process description supported in a representative business process management system, ObjectFlow [HK96].

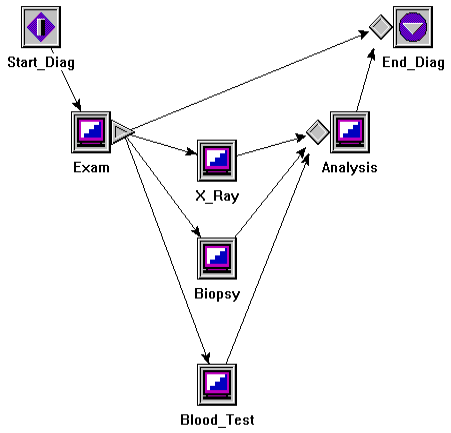

Figure 1: An example graphical representation of a process description

Instead of representing a process as a flow diagram some systems simply use a collection of rules to represent the process without explicitly delineating the paths. Such systems are generally more flexible in capturing the dynamic behavior.

Regardless of whether the process is described as a flow diagram, as a collection of rules, or some combination, the life cycle of automating a business process starts with a business analyst defining the process as a structured collection of abstract steps. The process description is further annotated with applications to be executed at each step. This stage most likely involves application development and/or wrapping of existing applications, and it produces executable process descriptions that can then be *registered* with a process management system. During execution time, an event triggers the process management system to create an instance of the process; the process management system then coordinates the step execution, and monitors and records the history. A business analyst, in turn, analyzes the history of process execution, potentially leading to improved definitions.

The long-running and distributed nature of a business process poses a challenge in enforcing transaction semantics over the entire process. Extending transaction models to support business processes has attracted considerable amount of research attention; this subject is further discussed in Section 4.

Not all business processes are naturally supported by process management systems. It is useful to distinguish business processes from two dimensions: task automation and process structure (Figure 2). Tasks in a process may be application centric (i.e., performed automatically by an application, typically some component of an ERP system or a software agent), or human centric (i.e., manual tasks involving human judgment, manual information gathering, or manual processing of desktop documents). The process can be highly-structured (the business rules and sequences that the tasks follow are predetermined and pre-scripted), semi-structured (parts of the rules are pre-scripted, parts of the rules may be modified or scripted on the fly - often referred to as an *ad hoc* process), and unstructured (there do not exist repeatable patterns of rules or sequences among the tasks, and participants often need to meet at the same time to perform work.) The design center for process management systems lies in the upper right-hand shaded corner, i.e., they are intended for managing processes composed of more application-centric tasks and are more structured, although the business processes they support often contain both manual and automated tasks, and they often accommodate certain degree of ad hoc scripting. In contrast, the lower left corner has been better supported by capabilities such as on-line shared space systems (e.g., Lotus Notes), or on-line meeting systems. One of the challenges in process management has been to provide end-to-end support for business processes that span both design centers: for example, a procurement process spans strategic sourcing, which tends to be less structured and involves human manipulation of documents, and order processing and payment processing, which tend to be more structured and involve mostly automated steps.

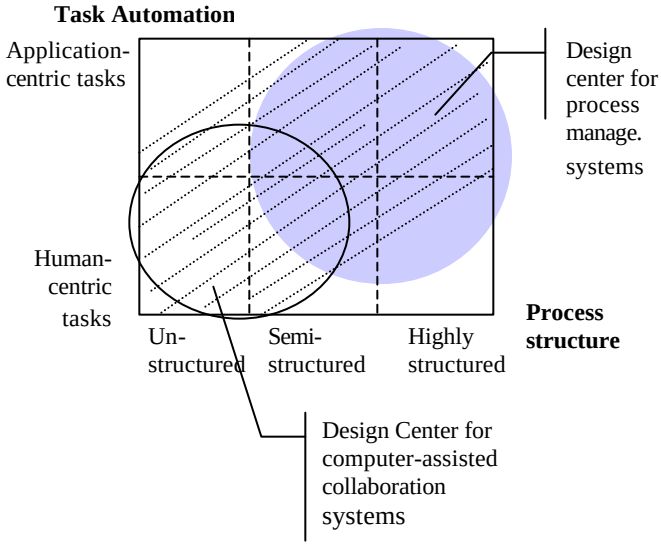

Figure 2: Framework for systems supporting business process management

2.2 Historical Perspective

Process management systems are often traced back to early office automation systems, document management systems, and workflow systems. [HK96] characterized the historical progression of the movement towards *open process management* as *homegrown workflow* (up to 1980's), where the systems were monolithic in nature with all information and control flow hard-coded in the applications; *object-routing workflow* (late 1980's and

early 1990's), where flexible scripting of workflow was offered in specific application packages such as document management or office automation systems, and *open architected process managers* (starting in the early to mid 1990's) where generic and open process management engines provide an infrastructure for an enterprise to integrate applications, data, and procedures from disparate systems and organizations. Open architected business process systems combine predefined work flow with ad hoc changes, use database and repository technologies for information sharing and persistence, use middleware technology for notification, distribution, and application invocation, and take advantage of object oriented technologies to provide customization. They focus on means for optimizing resources, enforcing policies, and providing monitoring and audit trail services.

The movement towards an architected process management infrastructure that started in the early 1990's has resulted in a number of product offerings (e.g. Action Technology's Action Workflow [MWF93, Dunham91], Xsoft's InConcert (now acquired by Ticbco) [MS93], DEC's ObjectFlow [HK96], IBM's FlowMark [LA93] (and subsequently, MQSeries Workflow), HP's Changengine (now called Process Manager [CS00], [HP]). The Workflow Management Coalition [WfMC], founded in 1993, was the first industrial consortium aimed at promoting frameworks and interoperability for open architected process management. It published a reference model and a set of associated standards.

This generation of product offerings coincided and amplified the wave of Business Process Reengineering (BPR) practice [HC93] that swept corporations in the early to mid-1990's. While many of the products have met commercial success, they, like BPR, have predominantly focused on *intra-enterprise* business process improvement and automation. The industry has not been as successful in driving pervasive standards in process-flow infrastructure as it has been in some other areas, such as message queues and object transaction management. The shift towards *inter-enterprise* business processes in the late 1990's has inevitably created new perspectives and challenges, as well as research and commercial opportunities.

2.3 Inter-enterprise (B2B) Collaborative Business Process Management

The potential business value of streamlining *inter-enterprise* business processes has fueled a renewed interest in process management technologies. However, the conventional intra-enterprise process management architecture faces a number of challenges. First of all, there must exist a mechanism to allow participating enterprises to reach agreement on the business process description and the data to be exchanged during process execution. To allow *scalable* B2B interoperation and alleviate the burden of pair-wise negotiation of integration points, a library of common, standard processes must exist so that by binding to a common, standard process, an enterprise achieves the ability of collaborating with a large number of partners' processes. Second, the process management function needs to be carried out as a collaboration among multiple distributed process managers. In essence, in crossing enterprise boundaries, the technologies traditionally suited for central coordination and integration need to be fundamentally reworked. The CrossFlow project is one research effort aimed at collaborative business process management [CrossFlow].

In [CH01], a *collaborative process framework* is proposed to extend the centralized process management technology (Figure 3). A collaborative process involves multiple parties, each playing a *role* in the process. Two aspects distinguish the collaborative process model from the conventional centralized process model:

1. The process definition is based on a commonly agreed business interaction protocol, such as the protocol for on-line purchase or auction.
2. The process execution is not performed by a centralized workflow engine, but by multiple engines collaboratively.

Common Definition of Collaborative Processes

The starting point of the collaborative process framework is the common definitions of collaborative business processes. There are at least 3 enablers for effective use of common business process definitions:

- there must exist a common business process meta-model and its associated schema language, so that business processes can be codified in a standard way;
- there must also exist a mechanism for common process descriptions, including business documents associated with these processes, to be easily re-used as building blocks for more complex processes, or customized for the special needs of vertical industries or geographical segments;
- finally, there must exist a mechanism for enterprises to publish their ability to participate in specific roles of common business processes as *process flow-enabled web services*, so that potential partners can automatically discover each other and engage in the process execution.

Several vendor- or consortium-based web service frameworks are emerging (e.g., World Wide Web Consortium [W3C], ebXML[ebXML], RosettaNet [RN], Open Buying Internet [OBI], Microsoft .Net/BizTalk [.Net], UDDI [UDDI], HP E-Speak [E-Speak]). RosettaNet's PIP (Partner Interface Process) is one of the earlier frameworks that made a significant contribution to the notion of business process-based e-commerce. The ebXML consortium, which is particularly interesting because it attempts to leverage the pre-existing industry experience in Electronic Data Interchange (EDI) in

designing new XML-based B2B framework, has several has several efforts that attempt to address the 3 issues listed above.[1] In particular, it is working on the *ebXML Business Process Specification Schema*, as a proposal to a standard business process meta-model. This model attempts to unify *document flow*, sometimes also referred to as *document choreography*, with *process collaboration*, which composes of document flows. Its proposal for XML *Core Components*, which addresses the issue of reusability of XML business documents, can potentially be extended to address the issue of reusability of business process schemas. Finally its *Registry & Repository* proposal specifies how to access a registry of information covering, for example, Business Partners, Business Services, and XML Schema definitions.

Collaborative Execution

In the collaborative process framework, the common business process definitions drive the definition and development of a *process-compliant service* at an enterprise. This is illustrated in the two Role Spec boxes in Figure 3. An enterprise determines the role(s) it wishes to play in a process, and develops the *role process specifications* and the corresponding internal execution control, including invocation or dispatching of *local services* (e.g. wrapped legacy applications). Once the process-compliant role specification is developed, it can be published as a web service. Note that there is no requirement that *local services* be published as web services, i.e., such services may not be directly accessible from trading partners. Local services are accessed indirectly through the local collaborative process manager.

As shown in Figure 3, each execution of a collaborative process, or a *logical process instance*, consists of a set of *peer process instances* run by the Collaborative Process Managers (CPMs) of participating parties. These peer instances conform in behavior to the specification of the role set forth in the common process definition, but may have private process data and sub-processes. The CPM of each party is used to schedule, dispatch and control the tasks that that party is responsible for, and the CPMs interoperate through an *inter-CPM protocol* to exchange the process data (or documents) and to inform each other on the progress of the process execution.

The framework requires that local CPMs interoperate with one another; however, there is no requirement that each local CPM be identical. To the extent that local CPMs are capable of enforcing the common business process specifications, they can differ significantly in functionality, such as support for internal data flow, local service integration, and nested sub flows. Therefore it is possible that, for example, one CPM is based on Microsoft's BizTalk Server [BizTalk], another is based on IBM's MQSeries ([IBM]) and WSFL (Web Service Flow Language) server ([WSFL]), and yet others can be based on NetFish's Process Manager ([pHub]), the APEx process engine ([APEx], or HP Process Manager [HP]. The Businesss Process Management Initiative ([BPMI]) intends to complement the public process interface standards, such as what ebXML's BPSS and RosettaNet's PIP strive to be, by providing a standard way to describe their private implementations.

Many challenging research issues remain in inter-enterprise collaborative business processes. They include: the ability to express and support transactional semantics, exception handling, and quality of services in the common process description; methods and tools conducive to achieving *a library* of reusable and composable common business documents and processes; efficient registration and searching/matching mechanisms; efficient and reliable inter-CPM execution protocols; and end-to-end process monitoring and analysis in a distributed environment. We also need to examine the requirements of many inter-enterprise business processes that contain a significant amount of human interactions: how traditional collaboration tools that are built around intra-enterprise collaborations (i.e., within a firewall) can be extended to an inter-enterprise environment, and ho they can interoperate with inter-enterprise collaborative process management. While the standards organizaitons and software vendors have made a great deal of progress, most of these issues are still not well understood, and will require joint efforts on the part of the research communities, standards organizations, software vendors, and leading edge enterprise participants.

3. Business Process Implementation based on Publish/Subscribe and Messaging

Traditionally, process coordination was achieved by using reliable queues [BHM90, MQSeries, MSMQ, BEA]. Message Queuing technology enables applications running at different times to communicate across heterogeneous networks and systems that may be temporarily offline. Applications send messages to queues and read messages from queues. Queuing products usually provide guaranteed message delivery, routing, security, and priority-based messaging. They can be used to implement solutions for both asynchronous and synchronous scenarios requiring high performance. Queues were usually implemented by using files. To provide reliability and transactional semantics, queues products had to implement many of the capabilities that databases traditionally implemented. Consequently,

[1] EbXML is an 18-month effot jointly sponsored by the United Nations body for Trade Facilitation and Electronic Business (UN/CEFACT) and the Organization for the Advancement of Structured Information Standards (OASIS) to produce a global standard for e-commerce.

several vendors recognized the advantages to be gained by integrating databases and queues [Oracle, Sybase].

In the last decade, Publish/Subscribe (or Pub/Sub) became the preferred building block for complex process implementation and coordination. Publish/Subscribe is a simple, yet powerful, paradigm for implementing dynamic business processes over intranet and extranet networks. Through the use of information channels, Pub/Sub provides an easy facility for instantly disseminating business information and events to multiple destinations. Usually, events are classified through a set of subjects (also known as groups, channels, or topics). Publishers publish events, by tagging each event with an appropriate subject, over real time channels that are subscribed to by consumers. Pub/Sub supports coordination by facilitating asynchronous one to many dissemination of events.

An alternative to subject-based systems, known as content-based systems, allows information consumers to request events based on the content of published events. An event in this case has a schema and can be modelled as a tuple containing attributes; e.g., (company, price, volume). A subscription is then a predicate over these attributes; e.g., (company = "EMC", price >100 and volume > 1000). This model is considerably more flexible than the subject-based model since it does not require predefinition of channels. Providing content-based filtering of messages is very powerful but more complex to implement.

The pub/sub paradigm was proven to be very useful in implementing business processes in a dynamically evolving system where the publishers and subscribers had to be anonymous, and hence a business step need not be aware of its preceding or following steps. In the past decade, very powerful and flexible business process automation systems were built on top of this basic paradigm. TIBCO (through its Information Bus) [OPS93, Chan98], was the pioneer in the use of Pub/Sub to build loosely-coupled distributed applications initially for the financial market, with other commercial companies following suit [Vitria, SQL/MX, MQSeries, Oracle, Sybase, Informix, NSSQL]. Pub/Sub is used by many of the world's largest financial institutions, deployed in the top semiconductor manufacturer' factory floors, utilized in the implementation of large-scale tracking and routing systems like FedEx, Internet services like Yahoo, Intuit, and ETrade, and chosen by many of the world's leading corporations as the enterprise infrastructure for integrating disparate applications. Implementations of Pub/Sub systems vary in their capabilities to guarantee reliable delivery of notification, provide "at most once" semantics, support for subject or content based routing, achieve scalability and availability, and the extent of their transactional support.

Many processes require transactionally guaranteed delivery for those applications that must update databases, consume messages on one set of subjects, and publish messages on another set of subjects, all within properly bracketed atomic transactions. Transactions that have to access queuing and/or publish/subscribe resource managers and a SQL database system, are forced to pay a high performance penalty. All resource managers have to participate in an expensive two-phase commit protocol. Furthermore, their lack of integration does not allow the SQL compiler to optimize access to both notifications and SQL data. Consequently, vendors like Tibco as well as some SQL database vendors (e.g. Oracle, Sybase, Informix) have integrated transactional queuing and publish/subscribe services with database products.

While these implementations remove the need for the two-phase commit protocol, their implementations use special purpose objects for queues and publication channels. This prevents queues and publication channels from being accessed as part of SQL select and update statements. This also prevents the SQL compiler from optimizing access to notifications and SQL data.

The transactional queuing and publish/subscribe extensions added to NonStop SQL/MX [SQL/MX, KV99] are tightly integrated into the database infrastructure. The pub/sub extensions do not introduce any special objects. Applications access regular SQL tables and use SQL select statements to subscribe and/or dequeue notifications. Applications use SQL insert and update statements to publish events. The extensions remove the need for a two-phase commit protocol and allow the SQL compiler to optimize access to both notifications and normal SQL data. The implementation can accomplish powerful filtering based on the message content, possibly joined with other database tables, as well as fully leverage the fault-tolerance (i.e. process pairs) and scalability features (i.e. horizontal partitioning) of the database engine. The queuing and publish/subscribe services are made available through embedded SQL and ODBC.

4. Advanced Transaction Models for Business Processes

An important contribution that the database community has made to business process management is to marry transactional concepts with workflows. Quite early on, it was recognized that business processes contained activities that accessed shared, persistent data resources, and so had to be subject to transactional semantics. In particular, the properties of atomicity and isolation could be usefully applied to business processes (or at least to parts of a business process). For example, one might want to declare that the payment of an invoice and crediting the payee's account were part of an atomic unit of work.

However, it was also recognized that it would be overkill to treat an entire business process as a single ACID transaction. First, since business processes typically are of long duration, treating an entire process as a transaction would require locking resources for long periods of time. Second, since business processes

typically involve many independent database and application systems, enforcing ACID properties across the entire process would require expensive coordination among these systems. Third, since business processes almost always have external effects, guaranteeing atomicity using conventional transactional rollback mechanisms is infeasible, and may not even be semantically desirable. It became apparent that the existing database transaction models would have to be extended.

Several models for long-running transactions have been developed to allow the definition of ACID-like properties at the business process level and to handle task failures (see [Elmagarmid92], [JK97] for papers describing several of these models).

Perhaps the earliest of the long-running transaction models was the Saga model [GS97]. A saga was a chain of transactions that was itself atomic. Each transaction in the chain was assumed to have a semantic inverse, or *compensation*, transaction, associated with it. If one of the transactions in the saga failed, the transactions were rolled back in the reverse order of their execution; committed transactions were rolled back by executing their corresponding compensation transactions.

In the Activity-Transaction Model (ATM), we allowed long-running transactions to be both nested and chained [DHL91]. Nesting allowed concurrency within a transaction (so, for example, tasks that were triggered by some event occurring in a parent transaction could execute in concurrent subtransactions) and also provided fine-grained, hierarchical control for failure and exception handling.. The original nested transaction model of [Moss85], which supported only *closed* subtransactions, was extended to include also *open* subtransactions [WS92]. A closed subtransaction commits its results tentatively to its parent; these partial results are externalized only after the top (root) transaction commits, thus ensuring atomicity and isolation of the whole transaction. In contrast, open subtransactions sacrifice isolation by directly externalizing their results. In ATM, failure handling was hierarchical. When a subtransaction failed, its parent was notified, and the parent could decide to execute an exception handler and retry the failed child, execute an alternate (contingency) task, or propagate the failure up the hierarchy. Propagating failures required compensating already committed subtransactions. Some tasks could be defined as *vital*, which meant that their failure caused the failure of the transaction hierarchy.

Subsequent work extended the ATM model to allow subtransactions whose commit scopes were in between the two extremes of closed and open nested transactions (essentially, a subtransaction could commit to any ancestor) [CD96, CD97]. Failure handling was hierarchical: the highest ancestor that needed to be aborted was identified, and then the subtree rooted at this ancestor was compensated or aborted. The model allowed compensation and contingencies to be associated with different levels of the hierarchy. Thus, sometimes it might be preferable to compensate an entire subtree instead of compensating every subtransaction in it. A further extension of this model applied to the case of propagating failures from one transaction hierarchy to another (a situation that might occur if data dependencies are established between two transactions in the same business process or between two business processes) [CD97, CD00].

Ideas similar to those in ATM also exist in other transactional workflow models. For example, *ConTracts* provide an execution and failure model for long-lived transactions and for workflow applications [Reuter92, RSS97]. A ConTract is a long-lived transaction composed of *steps*, whose order of execution is specified by a *script*. Isolation between steps is relaxed, so that the results of completed steps are visible to other steps; to guarantee semantic atomicity, each step is associated with a *compensating step* (or sub-script, if the compensation is a complex process) that semantically undoes the effect of the step. ConTracts provide both forward and backward recovery to manage failures. Backward recovery is achieved by compensating completed steps, typically in the reverse order of their execution. Compensation may be *partial*, meaning that it is performed up to a point in the contract from where forward execution can be resumed, possibly along a path that is different from the faulty one.

The basic ideas of transactional bracketing of parts of a business process, attaching compensation and contingency activities to the activities of the process, declaring some of the activities to be vital (or critical), and defining points in the process up to which rollback occurs on failure, followed by forward execution (roll forward) permeate many of the transactional models that were subsequently invented (e.g., *WAMO* [EL95, EL98], WIDE [GVBA99], CREW [KR98]). These models typically differ in how much flexibility the process designer has in specifying the backward compensation and forward execution process.

Commercial Products and Standards

A few commercial process management systems offer process meta-models that allow the definition of transactional properties (e.g., InConcert [MS93]). The transaction models are very similar to the ones described in the previous section.

The *Exotica* project [AKAE94, AAEK96] described methods and tools to implement advanced transaction models on top of Flowmark (predecessor of IBM MQ Series [IBM]). The basic idea was to provide the user with an extended workflow model that integrates advanced transaction concepts. The user could define a compensating task for each task of the workflow. A preprocessor would then translate these specifications into plain FDL (Flowmark Definition Language) by properly

inserting additional "compensating" paths after each task or group of tasks, which are conditionally executed upon a task failure. In particular, it was shown how sagas and flexible transactions could be implemented in Flowmark.

In [KS00] a transactional model for HP Changengine (now called Process Manager) is presented. The model allows the definition of *Virtual Transaction* (VT) regions on top of a workflow graph. If a failure occurs during the execution of a task enclosed in a VT region, then all tasks in the region are compensated in the reverse order of their forward execution, until a *compensation end point* is reached. Then, the system can retry the execution (up to a maximum number of times), follow an alternate path, or terminate the entire process execution. The virtual transaction model also allows for different isolation levels for VT regions: *serializable* (needs shared locks for reads and exclusive locks for writes), *read committed* (like serializable, but releases shared locks after reading), *read uncommitted* (no locks needed for reads), and *virtual isolation* (get read locks and release after reading, and get write locks only at the end of the transaction to perform all the updates in one shot).

Microsoft BizTalk Server 2000 includes a set of components, called Orchestration Services, which support the design and execution of business processes [Roxburgh00]. BizTalk processes, called *schedules*, are described by a graph whose nodes represent exchanges of BizTalk messages, typically corresponding to invocation of COM+ components. Process designers in BizTalk can associate transactional properties to subgraphs in a schedule. Three types of transactions are provided: short-lived (SLT), long-running (LRT), and timed. BizTalk also includes support for different levels of isolation, analogous to those of HP Changengine.

Standards bodies and industry consortia are also engaged in efforts to define transaction models at the business process level, both for processes within an organization and for inter-organization processes. In particular, OASIS has formed a *Business Transaction Technical Committee* (BTTC), with the goal of defining a transaction protocol for business processes that span across organizational boundaries [BTP]. While the proposals introduced above aim at defining transactional semantics for business processes, BTTC aims at defining transactional semantics for B2B protocols, such as the RosettaNet standards. BTTC assumes that each party involved in a multi-party B2B interaction is responsible for supporting transactional properties for the internal business processes, and instead defines a coordination protocol to ensure that all or none of the involved parties "commit" the effects of the B2B interaction. This problem is similar to that of coordinating distributed transactions in database systems, although carried over to the context of business processes and long-running transactions.

5. Business Process Intelligence

Organizations automate their business processes with the objectives of improving operational efficiency, reducing costs, improving the quality of service offered to their customers, and reducing human error. However, apart from graphical tools for designing business processes, and some simulation tools for detecting performance bottlenecks, business process management systems today provide few, if any, analytic tools to quantify performance against specific business metrics.

There has been some research on defining formal Petri Net-based execution semantics for business processes (see, for example, [van der Alst98]). In these approaches, the process definition graph is translated into some form of Petri Net, enabling formal properties such as termination and freedom from deadlock to be proved. Other approaches use formal logic to specify and reason about workflows and transactions (see, for example, [Bonner99].) However, these approaches are focused on verifying process definitions, not on analyzing or optimizing process executions.

The emerging area of *business process intelligence* is aimed at applying data warehousing and data mining technologies to analyzing and optimizing business processes.

Business process management systems today collect a lot of data about process executions: They log significant events that occur during process execution, such as the start and completion times of activities, the input and output data of each activity, failure events or other exceptions, and the assignment of resources to each activity. The log data is typically stored in a relational database, which can be queried to produce basic reports such as the number of workflow instances executed in a given time period, the average execution time, resource utilization statistics, etc. Traditional graphical querying and reporting tools are used for this purpose. However, these tools are limited in the types of analysis supported.

A more promising approach is to build a business process data warehouse, which can be loaded with the log data suitably cleaned, transformed, and aggregated, and perhaps integrated with data from other data source. The data in the warehouse can then be analyzed using multidimensional (OLAP) analysis tools and data mining tools to answer questions that a business manager or process analyst may be interested in:

- *analyze* the performance and quality of resources (e.g., resources of type A take 50% longer than average to execute activity B when the activity is initiated on Friday afternoons)
- *understand* and *predict* exceptions (e.g., what factors are strongly correlated with missed deadlines or other violations of service level agreements)
- *discover* conditions under which paths or subgraphs of the process are executed, so that the process can be redefined

- *optimize* process execution time and quality through optimal configuration of the system, assignment of resources, and dynamic adaptation of the process.

Exactly how to answer these questions is a ripe area for research. Two accompanying papers in this conference describe challenges in designing a process data warehouse to support these kinds of analysis [BCDS01], and the application of data mining techniques for understanding, predicting, and preventing exceptions [GCDS01].

Another interesting area of research is *business process discovery*, where the goal is to learn the structure of a business process from workflow log data [AGL98]. This would be especially useful for semi-structured and unstructured business activities where the process is not defined *a priori*. Such a "process" is usually implemented via rules triggered by events such as the start and completion of tasks. In some situations, there may be a latent structure that occurs regularly and that can be discerned by analyzing the history of events and rules. This learned structure can then be scripted and implemented efficiently as a business process. For inter-enterprise business processes, in particular, it is very unlikely that a complete business process that spans activities in different enterprises will be easily defined and agreed to by all the participants. In such situations, it may be possible to learn the underlying process by analyzing the sequences of interactions among the participants. Although [AGL98] made a promising start in this direction, process discovery is very much an open research issue.

6. Summary

Business process integration and automation have become high priorities for enterprises to achieve operational efficiency. With the burgeoning of e-commerce, there is a renewed interest in technologies for coordinating and automating intra- and inter-enterprise business processes. IN this paper, we reviewed the progress that has been made in the last decade and the state of the art today, in research, commercial products and standards. We identified open research issues, especially in collaborative, inter-enterprise process management, transaction models for business processes, and in the emerging area of business process intelligence.

We could not possibly be comprehensive in our coverage of this vast area, so we have chosen to highlight the topics that reflect our own interests and ongoing research activities.

Acknowledgment

Umesh and Mei wish to thank our colleagues at HP, especially Fabio Casati (who provided much of the material on transaction models, and who is leading the work on business process intelligence), Qiming Chen (who did much of the work on extensions to ATM and on collaborative process models), and Ming-Chien Shan (who leads the Business Process Management team at HP Laboratories).

References

[AAEK96] G. Alonso, D. Agrawal, A. El Abbadi, M. Kamath, R. Gunthor, and C. Mohan. Advanced transaction model in workflow context. *Proceedings of the 12th International Conerence. on Data Engineering*, New Orleans, Louisiana, February 1996.

[AGL98] R. Agrawal, D. Gunopulos, and F. Leymann. Mining Process Models from Workflow Logs. *Proc. of the Sixth International Conference on Extending Database Technology (EDBT)*, Valencia, Spain, 1998.

[AKAE94] G.Alonso, M.Kamath, D.Agrawal, A.El Abbadi, R.Gunthor, and C.Mohan. Failure handling in large scale workflow management systems. Technical Report RJ9913, IBM Almaden Research Center, November 1994.

[APEx] http://www.exterprise.com/solutions/apex.htm

[BCDS01] A. Bonifati, F. Casati, U. Dayal, and M.-C. Shan. Warehousing Workflow Data: Challenges and Opportunities. *Proc. 25th Intl. Conference on Very Large Data Bases*, September 2001.

[BizTalk] http://www.microsoft.com/technet/biztalk/btsdocs/

[Bonner99] A. Bonner. Workflow, Transactions, and Datalog. Proc. Of the Intl. Conference on Principles of Database Systems. May 1999.

[BPMI] http://www.bpmi.org/index.esp

[BTP] Business Transaction Protocol. Information available from www.oasis-open.org/committees/business-transactions/

[CD96] Q. Chen and U. Dayal. A Transactional Nested Process Management System. *Proc. 12th Intl. Conf. On Data Engineering*, February 1996.

[CD97] Q. Chen and U. Dayal. Failure handling for Transaction Hierarchies. *Proceedngs of the 13th Intl. Conference on Data Engineering*, IEEE Computer Society, April 1997.

[CD00] Q. Chen and U. Dayal. Multi-Agent Cooperative Transactions for E-Commerce. *Proc. 7th Intl. Conference on Cooperative Information Systems*, Lecture Notes in Computer Science 1901, Springer Verlag, 2000.

[CH01] Q. Chen and M. Hsu. Inter-Enterprise Collaborative Business Process Management. *Proc. International Conference on Data Engineering* (ICDE), Germany, 2001.

[Chan] Arvola Chan. Transactional Publish / Subscribe: The Proactive Multicast of Database Changes. *Proceedings of the ACM SIGMOD Intl. Conference on Management of Data*, May 1998,

[CrossFlow] http://www.crossflow.org

[CS00] F. Casati and M.-C. Shan. Process Automation as the Foundation for E-Business. *Proc. 26th Intl. Conference on Very Large Data Bases*, September 2000.

[DHL90] U. Dayal, M. Hsu. R. Ladin. Organizing Long-Running Activities with Triggers and Transactions. Proc. ACM SIGMOD Intl. Conference on Management of Data, May 1990.

[DHL91] U. Dayal, M. Hsu. R. Ladin. A Transactional Model for Long-Running Activities. *Proc. 17th nt; Conference on Very Large Data Base*s, September 1991.

[Dunham91] Dunham R., "*Business Design Technology Software Development For Customer Satisfaction*", Proc. 24th Annual Hawaii International Conference on Systems Sciences, 1991.

[ebXML] http://www.ebxml.org/

[EL95] J. Eder and W. Liebhart. The workflow activity model WAMO. *Proceedings of the 3rd International Conference on Cooperative Information Systems* (CoopIs), Vienna, Austria, May 1995.

[EL98] J. Eder and W. Liebhart. Contributions to exception handling in workflow management. *Proceedings of the Sixth International Conference on Extending Database Technology*, Valencia, Spain, March 1998.

[Elmagarmid92] A.K. Elmagarmid (editor). *Database Transaction Models for Advanced Applications*. Morgan-Kaufmann, 1992.

[E-Speak] http://www.e-speak.net

[GS87] H. Garcia-Molina and K. Salem. Sagas. *Proc. ACM SIGMOD Intl. Conference on Management of Data*, May 1987.

[Gartner00] R. Casonato. *Application Integration: Making E-Business Work*. Gartner Report, September 2000.

[GCDS01] D. Grigori, F. Casati, U. Dayal, M.-C. Shan. Improving Business Process Quality through Exception Understanding, Prediction, and Prevention. *Proc. 25th Intl. Conference on Very Large Data Bases*, September 2001.

[GVBA97] P.Grefen, J.Vonk, E.Boertjes, P.Apers. Two-layer Transaction Management for Workflow Management Applications. *Proceedings of the Eight International Conference on Database and Expert Systems Administration*, Touluse, France, 1997.

[HC93] Hammer M., and Champy J., *Reengineering the Corporation, A Manifesto for Business Revolution*, HarperBusiness, Harper Collins Publishers, 1993.

[HK96] M. Hsu and K. Kleissner. "ObjectFlow: Towards an Infrastructure for Process Management," *Journal of Distributed and Parallel Databases*, Vol 4, pp. 169-194, April 1996

[HP] Hewlett-Packard. HP Process Manager Product Overview. May 2001. Available from http://www.hp.com/go/e-process

[IBM] MQSeries. http://www.ibm.com/

[JB96] S. Jablonski and C. Bussler. *Workflow Management: Modeling, Concepts, Architecture, and Implementation.* International Thomson Computer Press, 1996.

[JK97] S.Jajodia and L.Kerschberg (editors), *Advanced Transaction Models and Architectures*. Kluwer Academic Publishers, New York, 1997

[KR98] M.Kamath and K.Ramamritham. Failure handling and coordinated execution of concurrent workflows. *Proceedings of the 14th International Conference on Data Engineering*, Orlando, Florida, February 1998.

[KS00] V. Krishnamoorty and M.C. Shan. Virtual Transaction Model for Workflow Applications. *Procs of SAC'00*. Como, Italy. Mar 2000.

[KV98] J. Klein, R. Van der Linde Non Stop SQL/MX Transactional Queuing and Publish/Subscriber Services. Workshop on High Performance Transaction Systems, 1998.

[LA93] Leymann F., Altenhuber W., *Managing Business Processes As Information Resource*, ITL IRM Conference, October 1993.

[Moss85]. E. Moss. *Nested Transactions*. MIT Press, 1985

[MS93] D. McCarthy and S. Sarin. Workflow and Transactions in InConcert. *Bulletin of the Technical Committee on Data Engineering*, IEEE Computer Society, 16(2), 1993.

[MWF93] Medina-Mora R., Wong H.T., Flores P., ActionWorflow as the Enterprise Integration Technology, *IEEE Data Engineering, 16(2),* June 1993.

[.Net] http://www.microsoft.com/net/defining.asp

[OBI] http://www.ebxml.org/

[OMG] Object Management Group. CORBA services: Common object service specification. Technical report OMG , July 1998.

[OPS93] Brian Oki, Manfred Pfluegl, Alex Siegel, Dale Skeen. The Information Bus – An Architecture for Extensible Distributed Systems. *Operating Systems Review, 27(5),* Dec. 1993.

[Oracle] Oracle 8i. http://www.oracle.com/

[pHub] http://www.iona.com/products/b2bihome.htm

[Roxburgh01] Ulrich Roxburgh. BizTalk Orchestration: Transactions, Exceptions, and Debugging. Microsoft Corporation. Feb 2001.

[Reuter92] A. Reuter. ConTracts. In A. Elmagarmid, ed., *Transaction Models for Advanced Database Applications*. Morgan Kaufmann, 1992.

[RosettaNet] http://www.rosettanet.org/

[RSS97] A. Reuter, K. Schneider, and F. Schwenkreis. Contracts revisited. In S. Jajodia and L. Kerschberg, editors, *Advanced Transaction Models and Architectures*. Kluwer Academic Publishers, New York, 1997.

[SGJR97] A. Sheth, D. Georgakopoulos, S.M.M. Joosten, M. Rusinkiewicz, W. Sacchi. J. Wileden, A. Wolf. Report from the NSF Workshop on Workflow and Process Automation in Information Systems. *ACM SIGSOFT Software Engineering Notes, 22(1),* 1997.

[SQL/MX] SQL/MX User Manuel, Compaq Computer Corp.

[UDDI] http://www.uddi.org/

[van der Alst98] W.M.P. van der Alst. The Application of Petri Nets to Worflow Management. *The Journal of Circuits, Systems, and Computers 8(1)*, 1998.

[Vitria] www.vitria.com

[W3C] http://www.w3.org/

[WfMC] http://www.wfmc.org/

[WS92] G.Weikum and H. Schek. Concepts and Applications of Multi-level Transactions and Open Nested Transactions. In A. Elmagarmid, ed., *Transaction Models for Advanced Database Applications*. Morgan Kaufmann, 1992.

[WSFL] http://www4.ibm.com/software/solutions/webservices/pdf/WSFL.pdf

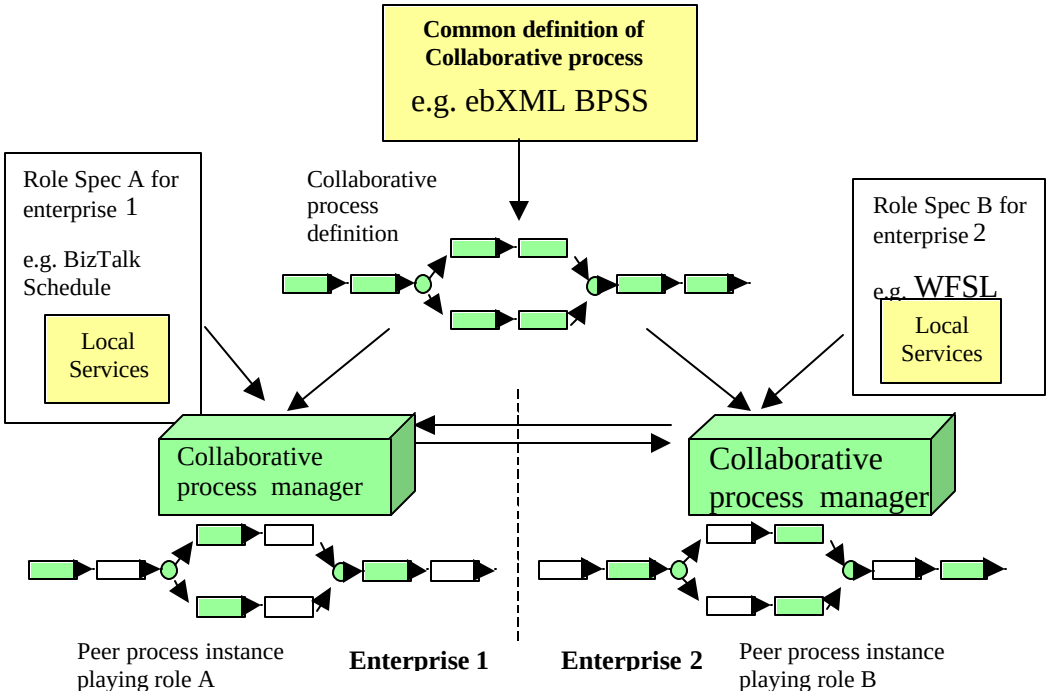

Figure 3: Collaborative Process Framework: Process Defintion and Execution